Environmental Monitoring and Pollution Prediction System

Scenario:

Students will develop an MLOps pipeline to monitor environmental data (e.g., air quality, weather, and pollution levels) and predict pollution trends.

Task 1: Managing Environmental Data with DVC

Objective: Use DVC to manage real-time environmental data streams collected from APIs.

Tasks:

1. **Research Live Data Streams:**

   Identify publicly available live data streams suitable for integration into the project. Examples include:

   - OpenWeatherMap API: Provides weather data, including temperature, humidity, and air pollution data (https://openweathermap.org/api).
   - AirVisual API (IQAir): Offers air quality data, including AQI and pollutant concentration (https://www.iqair.com/air-pollution-data-api).
   - EPA AirNow API: Provides air quality data from U.S. locations (https://docs.airnowapi.org/).
   - NOAA (https://pypi.org/project/noaa-weather/)

2. **Set Up DVC Repository**
   - Initialize a DVC repository for versioning collected data.
3. **Remote Storage Configuration**
   - Configure remote storage using free solutions like Google Drive or GitHub with DVC integration.

4. **Data Collection Script**
    - Write a Python script to fetch weather and air quality data. Fetch data at regular intervals.
5. **Version Control with DVC**
    - Use dvc add, dvc commit, and dvc push to manage and version the collected data.
6. **Automate Data Collection**
    - Schedule regular data fetching using cron jobs or task schedulers.
7. **Update Data with DVC**
    - As new data is fetched, update the data directory in the DVC repository by using dvc add to stage the new data files and dvc commit followed by dvc push to version and push the changes to the remote storage.

Deliverables of Task1:

- DVC repository containing environmental data.
- Automated data fetching script.
- Documentation of the integration process.

Task 2: Pollution Trend Prediction with MLflow

Objective: Develop and deploy models to predict pollution trends and alert high-risk days.

Tasks:

1. **Data Preparation**
    - Preprocess the environmental data (e.g., remove outliers, handle missing values).
2. **Model Development**
    - Use time-series models (e.g., ARIMA, LSTMs) to predict pollution levels or AQI trends.
3. **Train Models with MLflow**
    - Log experiments with MLflow, tracking metrics like RMSE, MAE, and accuracy.
4. **Hyperparameter Tuning**
    - Optimize models using grid search or random search techniques.

5. **Model Evaluation**
    - Compare models using appropriate metrics and choose the best one.
6. **Deployment**
    - Deploy the selected model as an API using Flask/FastAPI.

Deliverables of Task 2:

- Trained models with MLflow tracking.
- Deployed prediction API.
- Documentation of model training and deployment.

Task 3: Monitoring and Live Testing

Objective: Test the pipeline with live data and monitor the deployed system.

Tasks:

1. **Set Up Monitoring**
    - Use Grafana and Prometheus to track data ingestion, model predictions, and API performance.
2. **Test Predictions with Live Data**
    - Continuously fetch data from APIs to validate the deployed model's accuracy.
3. **Analyze and Optimize**
    - Analyze system performance and refine models or data pipelines as necessary.

Deliverables of Task 3:

- Real-time performance dashboard.
- Updates to the pipeline for optimization.
- Summary report on the system's live performance.